Event Documentation

5 October 2006

1 Log manipulation applications

There are two log-manipulating applications:

- Filterlog is a batch application that operates on disk files. It reads any collection of log files and filters their contents based on timestamp, event type, spacecraft time and arbitrary regular expressions. Output is written to a single disk file or to stdout.

- Copylog is an application that connects to a running ITOS and writes event messages to a disk file. The output file can be closed at regular intervals and a new file begun. An arbitrary shell command can be executed when a file change takes place. Messages can be filtered by event type and regular expression.

2 The filterlog application

Filterlog command-line arguments

Any string that is not part of any other argument is assumed to be the pathname of an input file. Stdin is used if no input file is specified.

Optional arguments:

-out <path>

Uses <path> as the output file. Stdout is used if no output file is specified.

-time<time1><time2>

Discards events with timestamps outside the range time1 to time2. If more than one -time argument is specified, the union of all ranges is used.
Times must be given in the format yy-ddd-hh:mm:ss

-event <spec>

Events in spec are selected. By default, all events are selected.
Example event specifications:

- "+RED_VIOL,+YEL_VIOL,+IN_LIMITS" selects those three events.
- "-NULL_EVENT, -STOL_ECHO" selects all events except those two.

Numbers can be used in place of event names. The following event group names can be used: "TM", "CMD", "STOL", "DSP" and "FTCP".

-exp <re> Discards messages from the previous event specification that do not contain regular expression <re>. The regular expression should usually be put into quotes to keep the shell from modifying it.

-sctime <time1> <time2>

Discards messages from the previous event specification that do not contain spacecraft times between <time1> and <time2>. Spacecraft times are included only in CFG_ALERT, SC_EVENT and telemetry events so it doesn't make sense to use -sctime with an event specification that includes any message types except those.
Times must be given in the format yy-ddd-hh:mm:ss

-help Writes a summary of command-line arguments and exits.

Discussion

If one or more -time arguments are used, messages are first filtered based on times. Messages outside the time intervals specified are not output. It does not matter where -time arguments are put on the command line.

The position of -event, -exp and -sctime arguments does matter. -exp and -sctime arguments modify the previous -event argument but do not affect any other -event arguments. For example

```
-event +CMD_EVENT -exp "telemetry" -event +CMD_VERIFY
```

will output CMD_EVENT messages only if they contain the string "telemetry", but will output all CMD_VERIFY messages.

Example filterlog invocations

```
filterlog /usr/tcw.wire/logs/startup_log -event "+CMD"
```

Writes all command events in startup_log to stdout. The command events are CMD_EVENT, CMD_VERIFY, CMD_MSG, CMD_WARN, CMD_ERROR, CMD_TF and CFG_ERROR.

```
filterlog /usr/tcw.wire/logs/startup_log -exp "file_capture"
```

Writes all events in startup_log containing the string "file_capture" to stdout.

```
filterlog /usr/tcw.wire/logs/startup_log -event "+TM" \
             -exp "tlmFrmSim" -event "+STOL" -exp "telemetry"
```

Writes all telemetry events that contain "tlmFrmSim" and all stol events that contain the string "telemetry".

```
filterlog -sctime 00-098-20:00:00 00-098-23:00:00
```

Reads from stdin and writes to stdout any telemetry, CFG_ALERT or SC_EVENT messages with spacecraft times between 20:00 and 23:00.

```
filterlog -sctime 00-098-20:00:00 00-098-23:00:00 -exp Assigning
```

Reads from stdin and writes to stdout any telemetry, CFG_ALERT or SC_EVENT messages with spacecraft times between 20:00 and 23:00 that contain the string "Assigning".

```
filterlog -event "+SC_EVENT" -sctime 098-20:22:00 098-23:00:00 -event "+CFG_ALERT"
```

Reads from stdin and writes to stdout SC_EVENT messages with spacecraft times from 20:22 to 23:00 and all CFG_ALERT messages.

```
filterlog -event "+SC_EVENT" -exp "this|that" -sctime 00-098-20:22:00 00-
098-23:00:00
```

Reads from stdin and writes to stdout SC_EVENT messages that contain one of the strings "this" or "that" and have spacecraft times from 20:22 to 23:00.

3 The copylog application

Copylog command-line arguments

-out <path>
Specifies the output path. If multiple files are produced (see -cut below), numbers are appended to <path>.
-out is **required**.

-cut <min>
Means a new output file is started every <min> minutes. Output file names are generated by appending a period and a number to the path specified in -out.

-exec <cmd>
Means <cmd> is executed immediately after each new output file is created. The name of the file just closed is appended to <cmd>.

-host <host>
Specifies the host from which events are obtained. The default is localhost

-port <num>
Specifies the port number on host from which events are obtained. The default is port 6066

-event<spec>
Events in <spec> are selected. By default, all events are selected. Example event specifications:

- "+RED_VIOL,+YEL_VIOL,+IN_LIMITS" selects those three events.
- "-NULL_EVENT, -STOL_ECHO" selects all events except those two.

Numbers can be used in place of event names. The following event group names can be used: "TM", "CMD", "STOL", "DSP" and "FTCP".

-exp <re> Discards messages from the previous event specification that do not contain regular expression <re>. The regular expression should usually be put into quotes to keep the shell from modifying it.

The copylog application exits when ITOS exits.

Example copylog invocations

```
copylog -out log
```

Simply creates a second log file named "log" in the current directory. That log file is closed when ITOS exits.

```
copylog -out log -event TM -exp violation
```

Creates a log file named "log" that contains only telemetry messages that contain the word "violation".

```
copylog -out log -cut 15
```

Puts the first fifteen minutes' worth of messages into a file named "log.1". The next fifteen minutes' worth are put into "log.2", and so on.

```
copylog -out log -cut 15 -exec dolog
```

As in the previous example, the first fifteen minutes' worth of log messages are put into file "log.1". Immediately after that file is closed, the shell command "dolog log.1" is executed. Fifteen minutes later log.2 is closed and "dolog log.2" is executed.